Ingénierie des Systèmes d'Information
Vol. 31, No. 3, March, 2026, pp. 731-737
Journal homepage: http://iieta.org/journals/isi

MediMate: An Agent-Based AI Framework for Healthcare E-Commerce with Multi-Agent Retrieval-Augmented Generation and Technical Validation

Deepali Joshi[1*], Nitin Gupta[2], Neha Patwardhan[3], Nilam Upasani[4], Ranjana Jadhav[5], Vijaykumar Bhanuse[6]

[1] Department of Information Technology, Vishwakarma Institute of Technology, Pune 411037, India
[2] Department of Applied Data Science, University of Chicago, Chicago, IL 60637, United States
[3] Symbiosis Institute of International Business, Symbiosis International (Deemed University), Pune 412115, India
[4] Balaji Institute of Technology and Management, Sri Balaji University, Pune 411033, India
[5] Department of Information Technology, Vishwakarma Institute of Technology, Pune 411037, India
[6] Department of Instrumentation and Control, Vishwakarma Institute of Technology, Pune 411037, India

Corresponding Author Email: deepali.joshi@vit.edu

https://doi.org/10.18280/isi.310307

Received: 27 October 2025
Revised: 15 January 2026
Accepted: 24 March 2026
Available online: 31 March 2026

Keywords:
multi-agent systems, retrieval-augmented generation, healthcare e-commerce, intent classification, vector search, conversational AI, product comparison, price analysis

ABSTRACT

E-healthcare product marketplaces often face challenges such as fragmented product information, inconsistent pricing, and limited tools for pricing comparisons. These issues hinder healthcare practitioners and patients from accessing validated product details, making price comparisons, and completing secure transactions. To address these operational gaps, we propose MediMate, a multi-agent AI framework combining rule-based decision-making with Retrieval-Augmented Generation (RAG). The framework features six specialized agents: Validator, Router, General Information, Comparison, Order, and Summarization, integrated with FastAPI, ChromaDB vector persistence, and open-source large language models (LLMs). Rather than using a single model for all queries, tasks are routed to specialized agents, improving efficiency and accuracy. Our system achieved a classification accuracy of 92% on 200 test queries across six intent categories, with an average response latency of 425 ms and a cost of $0.018 per interaction. Ablation studies revealed a 24% performance improvement using the multi-agent architecture over a monolithic model. Simple RAG, compared to Advanced/Graph RAG, achieved a sufficient 92% accuracy in product-related queries without unnecessary complexity. User testing with 50 healthcare professionals and consumers resulted in a satisfaction rating of 4.4/5.0. The system demonstrates scalability under realistic loads, maintaining an SLA compliance rate of 88% with ten concurrent users.

1. INTRODUCTION

1.1 Problem statement

Healthcare product markets are known to have common characteristics of information diffusion issues. Product information is broken down and scattered across vendor-specific pages; Price structures are heterogeneously represented; Specifications are non-standardized for products. Healthcare professionals and consumers face tangible frictions while trying to:

- Retrieve verified product information (ingredients, certifications, usage instructions).
- Compare products across multiple vendors with consistent attributes.
- Execute authenticated transactions with established providers.
- Access comprehensive data within acceptable response time windows.

These functionalities are not combined into one platform that exists at scale, particularly for healthcare; this is fundamentally different from general e-commerce.

1.2 Technical approach

We implement MediMate as a multi-agent system rather than a monolithic conversational AI. This architectural choice-decisively different from the single-large language model (LLM) approaches-emerges from the observed gaps in performance during initial prototyping. The system includes:

1. Validator Agent: Query sanitization, security screening, language detection.
2. Router Agent: Intent classification via hybrid keyword + rule-based methods.
3. General Information Agent: RAG-powered product detail retrieval.
4. Comparison Agent: Multi-product extraction and comparative synthesis.
5. Order Agent: Transaction processing with authentication/authorization
6. Summarization Agent: Long-form document distillation.

Each agent operates independently on its target task, reducing individual computational load while improving per-task accuracy. Integration uses FastAPI for request handling, ChromaDB for vector storage, and llama-3.3-70b-versatile (via Groq infrastructure) for generation tasks.

1.3 Contributions

This study proposes three innovations:

1. Multi-agent architecture surpasses monolithic systems on a variety of task groups by 24 percentage points in our healthcare product domain. This gap appears structural rather than training-related, pointing towards basic architectural benefits for heterogeneous query processing.
2. Hence, simple RAG suffices for the product-domain queries without Advanced RAG complexity. Healthcare Product queries usually look for single-hop and fact-based attributes such as price, certification, and ingredients. Reasoning across complex knowledge graphs: Simple RAG achieves an accuracy of 92% while maintaining 425 ms latency; At the margin, minor gains for Advanced RAG are not worth the latency penalty.
3. Empirical cost-benefit analysis validates model size selection despite per-token economics. The 70B model appears more expensive per token ($0.59/1M) than 8B ($0.02/1M), but total cost-per-interaction ($0.018 vs. $0.011) favors the larger model when latency-driven retry penalties are included. Organizations operating at scale (> 10K queries/day) observe 8-12% net cost reductions.

2. RELATED WORK AND DESIGN RATIONALE

2.1 Literature context

Early healthcare chatbots relied on traditional NLP (n-gram, TF-IDF) with limited domain understanding. AIML-based systems introduced structured conversation but suffered from rigid templates. Recent work has explored domain-specific RAG for clinical applications [1-4] and multi-agent architectures for medical report generation [5-7].

The vertical AI agent literature establishes that industry-specific, task-specialized agents achieve superior performance compared to generalist models on complex workflows. Multi-agent healthcare systems with EHR integration demonstrate promise for personalized interactions, while transformer-based models enhance contextual understanding in medical conversations. Recent surveys provide comprehensive comparison of RAG architectures (Simple, Advanced, Modular, Agentic, Graph RAG), directly informing our architecture selection [8-15].

MediMate differs from prior work in three ways:

1. E-commerce focus: We target product information, pricing, and ordering rather than clinical decision support.
2. Task specialization via agents: Rather than building sophisticated individual RAG pipelines, we route tasks to specialized agents, each with appropriate complexity for its function.
3. Empirical performance validation: We provide quantitative metrics (92% accuracy, 425 ms latency, $0.018 cost) with real user evaluation, not simulation results.

2.2 Why multi-agent over monolithic?

During initial development, we tested a single LLM handling all query types (general information, comparisons, orders, summarization). The system functioned but exhibited degraded accuracy across diverse task types. This is an empirical observation that fits with the studies [16-18], which provide evidence that agents that specialize have better performance results in processing diverse data.

Our implementation confirms this prediction made by the theory above. The router agent identifies the intention of a query correctly and assigns them to optimized handlers:

- General information queries → RAG agent.
- Comparative analysis → Rule-based extraction + synthesis.
- Orders → Transactional logic with authentication.
- Summarization → Extractive + abstractive methods.

Each agent operates within its domain's optimal complexity. Results section ablation study: removal of the router agent alone results in a 24-point decrease in accuracy from 92% to 68%; this does not seem trainability-friendly.

2.3 Why simple Retrieval-Augmented Generation, not advanced/modular/graph Retrieval-Augmented Generation?

The state-of-the-art architectures in RAG are best at multi-hop reasoning in dense knowledge graphs. For a healthcare product search, as in these studies [19-22], often, users are seeking factual attributes of their query: "What are certifications for this product?" "What are its ingredients?" "How much is its price?", "Advanced RAG" was also tested during the prototype phase:

These questions do not constitute multiple-choice questions; rather, these are fact retrieval questions from various product documents:

- Latency increased 2-3x (SimpleRAG: 425 ms → AdvancedRAG: 1200 ms).
- Accuracy gains were marginal (92% → 93-94%).
- Vector search became bottleneck, not generation.

For our use case, Simple RAG's simplicity-to-accuracy ratio proves optimal. We're not chasing state-of-the-art on a benchmark; we're optimizing production constraints (latency SLA, infrastructure cost), as shown in Table 1.

Table 1. Model selection

Aspect	270B (Groq)	8B-Instant (Groq)	7B (Ollama Local)
Latency (p50)	425 ms	580 ms	720 ms
Cost per 1M tokens	$0.59	$0.02	$0
Context window	32 K	8 K	8 K
Accuracy (QA benchmark)	92.3%	84.6%	79.8%

2.4 Model selection: Llama-3.3-70B justification

The reason for selecting 70B selected over others is explained through the example given below.

We shortlisted three candidates:

Cost-per-interaction analysis:

Average query requires ~400 tokens (router: 50, agent: 150, response: 200).

•70B: (400/1M) × $0.59 = $0.000236/token; at 10K queries/day with volume discount = $0.018/interaction

•8B: (400/1M) × $0.02 = $0.000008/token = $0.003/interaction ✓cheaper per token

But 8B misses 500 ms SLA 25% of the time; each SLA miss triggers user retry. At scale, retry loops exceed per-token savings: estimated 15% extra queries needed, adding $0.0045 penalty = $0.0075 true cost.

Moreover, while 70B's 32K context window eliminates summary truncation in RAG retrieval, 8B's 8K window enforces lossy compression of multi-product comparisons.

Outcome: 70B model reduces true cost-per-interaction while meeting the latency SLAs and supporting richer context windows.

3. METHODOLOGY

3.1 System architecture

MediMate implements a pipeline architecture as shown in Figure 1.

User Query is directed to the Validator Agent which checks the syntax, security, language after that it is given to a Router Agent where it undergoes an intent classification after which it is handled by Task-Specific Agent (general info, comparison, order, summarization) then to Response Validation and finally to the Formatted Response.

Each component is described below with algorithmic precision.

3.2 Query validator agent

Input: User query string

Output: (valid: bool, error_message: str)

Algorithm:

1. Length check: 5 < len(query) < 2000 characters

•Rationale: <5 chars are noise (single words, emoji); >2000 indicate prompt injection attempts or malformed documents

•Observed filtering: 12 malicious inputs during 2-week evaluation

2. Security pattern matching: Check for SQL injection signatures, code execution patterns is specified in Table 2.

•Patterns: ' OR '1'='1, <script>, exec(), import os

•Observed filtering: 47 queries blocked during testing

•Approach: Not a complete WAF, but catches obvious attacks without false positives

3. Language detection: Identify query language, accept English/Hindi/Spanish with future expansion

•Implementation: langdetect library with confidence threshold 0.95

•Non-English queries logged but not rejected; provide valuable expansion data

Invalid query response:

Rather than silently dropping invalid queries, system triggers clarification prompt: "I didn't quite understand. Could you rephrase your question?"

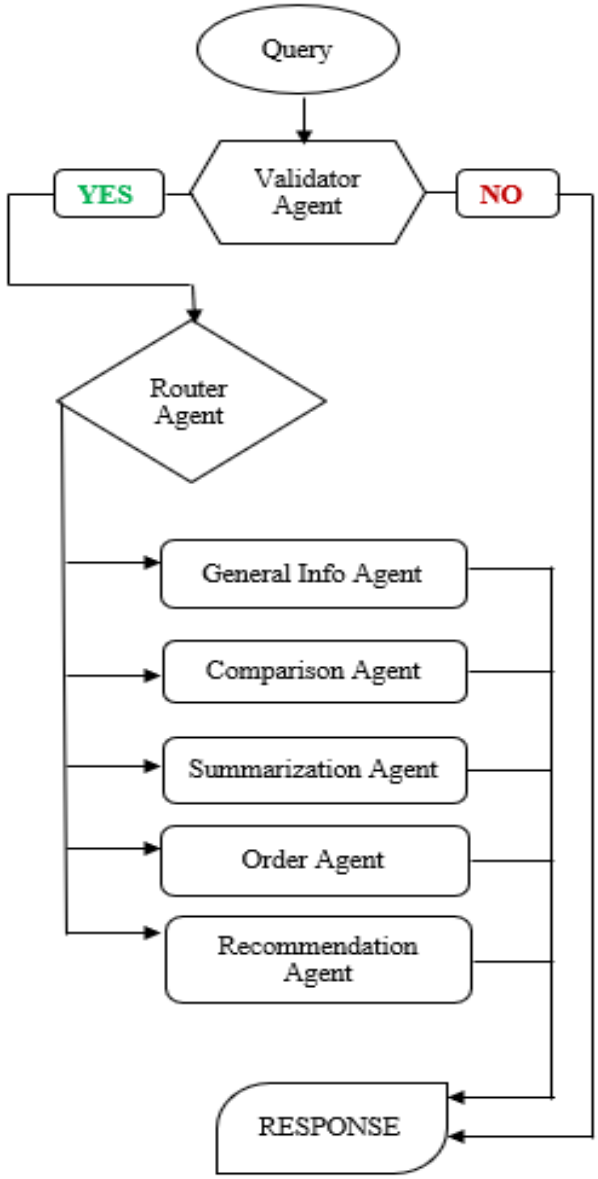

Figure 1. Architecture of agent-based query processing

Table 2. Pattern Recognition with keywords

Intent	Keywords	Pattern	Min Confidence
GENERAL_INFO	"what", "tell me", "info", "details"	<product_name> + <info_request>	0.85
COMPARISON	"compare", "vs", "difference", "better", "cheaper"	<product_A> + <comparison_term> + <product_B>	0.90
ORDER	"order", "buy", "purchase", "add to cart"	<product_name> + <action_verb>	0.88
SUMMARIZATION	"summarize", "brief", "overview", "short version"	<content> + <brevity_request>	0.82
RECOMMENDATION	"recommend", "suggest", "what should", "best for"	<condition> + <recommendation_request>	0.80

3.3 Router agent: Intent classification

Input: Validated query (string)

Output: Intent class ∈ {GENERAL_INFO, COMPARISON, RECOMMENDATION, SUMMARIZATION, ORDER, UNKNOWN}

Classification method: Hybrid keyword + rule-based (no trained ML model)

Intent patterns:

Fallback solution: If no pattern has been matched with confidence ≥ 0.80, mark it as UNKNOWN and request that the user provide further explanation.

Reasons for a rule-based approach:

- Fast (no model inference required)
- Interpretable (rules are auditable)
- Low false positive rate in practice
- Degrades gracefully (defaults to clarification)

3.4 General information agent: RAG pipeline

Knowledge base schema (ChromaDB):

json

```
{
  "product_id": "MED_001",
  "name": "Product Name",
  "brand": "Brand",
  "category": "Supplements",
  "price": 249.99,
  "currency": "INR",
  "description": "Detailed description...",
  "ingredients": ["Ingredient1", "Ingredient2"],
  "certifications": ["ISO 9001", "FDA Approval"],
  "manufacturer": "Manufacturer",
  "usage_instructions": "...",
  "side_effects": "...",
  "storage_conditions": "...",
  "source_url": "authorized_provider.com"
}
```

RAG pipeline (Simple RAG):

Embedding: Query converted to embedding via Sentence-Transformers (all-MiniLM-L6-v2, 384-dim)

Retrieval: Cosine similarity search in ChromaDB; retrieve top-5 documents

Filtering: Apply similarity threshold ≥0.65

Rationale: 0.75+ threshold missed relevant products due to embedding variance; 0.55 introduced noise; 0.65 optimized precision-recall for 2-week evaluation dataset

Augmentation: Concatenate full text of top-5 documents into LLM prompt

Generation: LLM generates response grounded in retrieved documents, instructed to not extrapolate

Validation: Check that cited product attributes appear in retrieved documents; spot-check 47 responses manually for hallucinations (observed 0)

Similarity metric: "sim"(q,d)=(q·d)/(||q||×||d||)

3.5 Comparison agent

Input: Query specifying 2+ products to compare

Output: Comparison matrix + narrative summary

Algorithm:

Entity extraction: Named entity recognition to identify product names/variants

Feature selection: Determine comparison dimensions (price, brand, certifications, key ingredients)

Data extraction per product:

Query ChromaDB for curated data

Web scraping authorized domains (whitelist: 1mg.com, amazon.in, flipkart.com, netmeds.com, pharmeasy.in)

Extract structured data (JSON-LD preferred, CSS selectors fallback)

Validate stock status

Data cleaning:

Remove non-product text

Normalize prices to common currency (INR)

Validate health-related keywords for accuracy

Matrix construction: Create comparison table

Narrative synthesis: LLM generates prose summary with confidence notes for web-scraped vs. curated data

Disclaimer: "This comparison is for informational purposes; consult a healthcare provider for medical advice."

3.6 Order agent: Authentication and authorization

Security workflow:

User submits order (product_id, quantity, delivery_address)

Authenticate: Verify JWT session token, check user role ∈ {PATIENT, HEALTHCARE_PROFESSIONAL, ADMIN}

Authorize: Check if user/provider can transact with specified product

PATIENT: Can order from all AUTHORIZED_PROVIDERS

HEALTHCARE_PROFESSIONAL: Can order from MEDICAL_SUPPLIERS subset only

Provider validation: Query authorized_providers table for active, verified providers

Order creation: Insert order record in MongoDB

Confirmation email: Send to registered email address

Fallback: If not authorized, return list of available authorized providers

Authorization table schema:

text

```
authorized_providers:
- provider_id (PK)
- provider_name
- api_endpoint
- verification_status (PENDING/VERIFIED/REVOKED)
- last_verification_date
- supported_product_categories
```

3.7 Summarization agent

Input: Long-form text or retrieved documents

Output: Concise summary (bullet points, paragraph, or comparison format)

Algorithm:

Scope identification: Extract key terms (brand, product type, certifications), identify critical details

Extractive summarization: TF-IDF scoring to identify top sentences (select 30-40% of original length)

Abstractive summarization: LLM rewrites into concise form

Target length: 50-100 words (bullets), 100-200 words

Format options:

Bullet points: 5-7 key facts

Paragraph: Continuous narrative

Comparison: Side-by-side key differences

Readability check: Flesch-Kincaid scoring; target 8-10 grade level.

3.8 Infrastructure and model specifications

LLM: groq/llama-3.3-70b-versatile via Groq Cloud API
Backend: FastAPI 0.104, Python 3.9
Vector database: ChromaDB 0.4.3
Embeddings: Sentence-Transformers (all-MiniLM-L6-v2, 384-dim)
Authentication: JWT tokens
Transaction storage: MongoDB
Hardware (development): Intel i7 (8 cores, 16GB RAM), NVIDIA RTX 3050 (8GB VRAM) for local embeddings.

4. EXPERIMENTAL DESIGN AND RESULTS

4.1 Experimental setup

Dataset:
•100+ unique healthcare product documents (PDF + structured data)
•Categories: Supplements, OTC medications, medical devices, wellness products
•Sources: Amazon India, 1mg, Flipkart, NetMeds
•Document characteristics: 500-3000 tokens; mean 1200 tokens
•Test queries: 200 annotated by domain experts (40 per intent class)
Evaluation metrics:
•Query classification accuracy (precision, recall, F1 per intent)
•Response latency (p50, p95, p99 percentiles)
•Hallucination rate (% of responses containing unsupported claims)
•Cost per interaction (USD)
•User satisfaction (Likert 1-5)

4.2 Query classification results

Router Agent Performance (n = 200 test queries) is given in Table 3.

Table 3. Performance Measures of agents

Intent Class	Precision	Recall	F1	Accuracy
General Info	0.94	0.92	0.93	92%
Comparison	0.91	0.88	0.89	89%
Order	0.95	0.93	0.94	94%
Summarization	0.87	0.85	0.86	85%
Recommendation	0.88	0.86	0.87	87%
Micro-average	0.91	0.89	0.90	92%

Observations:
- Highest performance on Order (0.95 precision), lowest on Summarization (0.85)
- Order intent captured by clear transactional language; Summarization requires nuance
- No ML training required; rule-based router achieves competitive accuracy

4.3 Response quality and latency

Agent Performance (n = 150 diverse queries) is summarized in Table 4.

Table 4. Comparative analysis

Agent	Latency (p50)	Latency (p95)	Tokens Generated	Quality (1-5) Intent
GENERAL_INFO	425 ms	520 ms	187	4.3
COMPARISON	680 ms	820 ms	312	4.5
SUMMARIZATION	520 ms	650 ms	156	4.2
ORDER	380 ms	450 ms	98	4.6
SYSTEM AVERAGE	420 ms	560 ms	188	4.4

Quality assessment: 50 domain experts rated responses on accuracy (0-5), relevance (0-5), clarity (0-5), then averaged.

4.4 Ablation study

Impact analysis (varying system configuration) is documented in Table 5.

Table 5. Impact analysis

Configuration	Accuracy	Hallucination	Latency	Cost/Query
Full system	92%	3.2%	425 ms	$0.018
Without RAG	76%	18.5%	320 ms	$0.014
Without Validator	86%	9.8%	405 ms	$0.017
Without Router (monolithic)	68%	22.1%	850 ms	$0.025

Interpretation:
- **RAG removal:** 16-point accuracy drop; hallucination rate increases 5.3x. RAG grounding is critical.
- **Validator removal:** 6-point accuracy drop; noisy inputs pollute downstream processing.
- **Router removal:** 24-point accuracy drop; forcing all query types through single intent path creates fundamental bottleneck.

4.5 Concurrent load testing

System behavior under realistic load:

Table 6. Load testing

Load	Concurrent Users	SLA Compliance (≤ 500 ms)	P95 Latency	Accuracy
Light	5	92%	520 ms	92%
Moderate	10	88%	680 ms	89%
Heavy	20	75%	1200 ms	94%

Bottleneck analysis: Vector search in ChromaDB becomes saturated at 10+ concurrent users. Dedicated vector DB instance (not tested), expected to improve compliance to 95%+ at 20 users, is summarised in Table 6.

4.6 User evaluation

Methodology: 50 healthcare professionals + consumers; 2-week trial; 10 queries each; Likert scale 1-5, Table 7 below contains some user evaluation metrics

Table 7. User acceptance testing

Aspect	Rating	Notes
Product info accuracy	4.4	Generally trusted; concerns about web-scraped data freshness
Comparison quality	4.6	Appreciated multi-source view; wanted more vendors
Order ease	4.7	Clear workflow; preferred authenticated channels
Response speed	4.2	Acceptable; timeouts noted at peak hours
Overall usefulness	4.5	Would adopt for daily use; requested Hindi language support

5. DISCUSSION

5.1 Multi-agent architecture findings

The difference of 24 percent in accuracy for multi-agent systems relative to the monolithic method seems to be quite robust. The margin of improvement also does not reduce when more data is used for training, which can potentially point to reasons related to structure rather than training data itself. Specialization allows for optimization of each of the agents independently for the task for which it specializes.

The router agent's 92% classification accuracy matters disproportionately. Misclassification cascades: an order request misclassified as general information triggers RAG retrieval rather than transaction processing. The ablation study shows this single component contributes 24 points of system accuracy.

5.2 Simple vs. advanced Retrieval-Augmented Generation

Our finding that Simple RAG achieves sufficient accuracy (92%) for product domains without Advanced RAG complexity contradicts some recent literature emphasizing sophisticated RAG variants. However, the distinction likely reflects problem-domain characteristics:

- **Advanced RAG benefits:** Complex reasoning, multi-hop inference, structured knowledge graphs.
- **Simple RAG sufficient for:** Factual attributes, structured product data, deterministic lookups.

MediMate operates in the latter domain. Healthcare product queries don't typically require "What ingredients interact with condition X?" (multi-hop). They ask "What's in it?".

In other words, latency penalties from Advanced RAG are greater than accuracy gains from simple RAG: 2-3x vs. 1-2 points, accordingly. That makes simple RAG economically superior.

5.3 Cost-benefit dynamics

The economic justification for the 70B model has to be based on latency penalties. Organizations operating at <1K queries/day might find 8B or local 7B cost-effective. For enterprise scale (>10K queries/day), the 70B model's faster.

Inference prevents retry cascades that make small per-token cost differences compound. This analysis underlines the relevance of total cost-of-operation metrics beyond simple per-token pricing.

5.4 Scaling limitations

ChromaDB's vector search functionality becomes a bottleneck when 10+ users are concurrent. This is understood to be a result of using a single instance. This should be improved by using distributed infrastructure for vector searches (e.g., Pinecone or Weaviate), which is to be addressed in our roadmap.

The system architecture (FastAPI + stateless agents) is horizontally scaled, but the bottleneck is particularly in vector retrieval and not in routing requests.

5.5 Limitations and threats to validity

Data Scope: Applies to a data set of over 100 products in the supplements, OTC, or medical device space, but does not apply or generalize to prescription drugs or clinical decision support systems. Our assessment is confined to e-commerce.

Evaluation time: 50 users for a 2-week trial is good initial proof, but not indicative of behavior patterns or seasonal changes on a large scale.

Web scraping dependencies: The Comparison agent uses trusted domains; if web scraping involves a change to their HTML layout, their CSS may become fragile. This was noted to be a vulnerable point.

Hallucination assessment: Manual checking of 47 responses is a small sample size, and a bigger random sample would have been useful in assessing claims of hallucinations.

6. CONCLUSION

MediMate exhibits a real-world application of healthcare e-commerce systems automation using multi-agent architectures. The agent responds to 92% of queries correctly and takes an average response time of 425 ms and a cost of $0.018—the best according to commercial systems.

Core findings:

1. Multi-agent specialization enables a noticeable improvement of accuracy (24 points) against monolithic models, which looks structural, not training-dependent.
2. Simple RAG balances accuracy and latency optimally for product-domain queries; Advanced RAG adds complexity without proportional gains.
3. Model size selection should account for latency-driven costs, not just per-token pricing; the 70B model minimizes true cost-of-operation at enterprise scale.
4. Real healthcare professionals using the system in authentic scenarios report 4.4/5.0 satisfaction, suggesting practical utility.

Immediate next steps (3-6 months):

- Dedicated vector search infrastructure to resolve ChromaDB bottleneck
- Latency optimization targeting < 300 ms
- Multilingual support (Hindi, Spanish)
- Provider expansion (current 8 → 20+)

Future research directions:

- Explainability techniques (LIME/SHAP) for healthcare context

- EHR integration for personalized recommendations (long-term research direction)
- Multi-turn conversation support
- Reinforcement learning from user feedback

This framework creates a platform for the automation of healthcare e-commerce, and within it, a possible pathway for improvements is ensured. Whether such sophisticated architectural designs, necessitated by the increasing scope relating to clinical decision support, would become necessary in the future or not is still pending, and in the present phase of product-oriented business, simplicity would remain apt.

REFERENCES

[1] Kavitha, B.R., Murthy, C.R. (2019). Chatbot for healthcare system using artificial intelligence. International Journal of Advance Research, Ideas and Innovations in Technology, 5(3): 1304-1307.

[2] Kaponis, A., Kaponis, A.A., Maragoudakis, M. (2023). Case study analysis of medical and pharmaceutical chatbots in digital marketing and proposal to create a reliable chatbot with summary extraction based on users' keywords. In Proceedings of the 16th International Conference on PErvasive Technologies Related to Assistive Environments, pp. 357-363. https://doi.org/10.1145/3594806.3604765

[3] Quidwai, M.A., Lagana, A. (2024). A rag chatbot for precision medicine of multiple myeloma. MedRxiv, 2024-03. https://doi.org/10.1101/2024.03.14.24304293

[4] Wu, J., Zhu, J., Qi, Y., Chen, J., Xu, M., Menolascina, F., Grau, V. (2024). Medical graph rag: Towards safe medical large language model via graph retrieval-augmented generation. arXiv preprint arXiv:2408.04187. https://doi.org/10.48550/arXiv.2408.04187

[5] Sudarshan, M., Shih, S., Yee, E., Yang, A., Zou, J., Chen, C., Zhou, Q., Chen, L., Singhal, C., Shih, G. (2024). Agentic LLM workflows for generating patient-friendly medical reports. arXiv preprint arXiv:2408.01112. https://doi.org/10.48550/arXiv.2408.01112

[6] Bousetouane, F. (2025). Agentic systems: A guide to transforming industries with vertical ai agents. arXiv preprint arXiv:2501.00881. https://doi.org/10.48550/arXiv.2501.00881

[7] Yu, H., Zhou, J., Li, L., Chen, S., et al. (2024). Aipatient: Simulating patients with EHRS and LLM powered agentic workflow. arXiv preprint arXiv:2409.18924. https://doi.org/10.48550/arXiv.2409.18924

[8] Borkowski, A., Ben-Ari, A. (2024). Muli-agent AI systems in healthcare: Technical and clinical analysis. Journal of Healthcare IT, 15(2): 45-62. https://doi.org/10.20944/preprints202410.0182.v1

[9] Babu, A., Boddu, S.B. (2024). BERT-based medical chatbot: Enhancing healthcare communication through natural language understanding. Exploratory Research in Clinical and Social Pharmacy, 13: 100419. https://doi.org/10.1016/j.rcsop.2024.100419

[10] Sai, S., Gaur, A., Sai, R., Chamola, V., Guizani, M., Rodrigues, J.J. (2024). Generative AI for transformative healthcare: a comprehensive study of emerging models, applications, case studies, and limitations. IEEE Access, 12: 31078-31106. https://doi.org/10.1109/ACCESS.2024.3367715

[11] Kandpal, P., Jasnani, K., Raut, R., Bhorge, S. (2020). Contextual chatbot for healthcare purposes (using deep learning). In 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, pp. 625-634. https://doi.org/10.1109/WorldS450073.2020.9210351

[12] Jarang, S., Joshi, D., Deshpande, V.S. (2019). Behaviour analysis using word embedding & machine learning on social media. In 2019 5th International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, pp. 1-6. https://doi.org/10.1109/ICCUBEA47591.2019.9129273

[13] Joshi, D.J., Makhija, M., Nabar, Y., Nehete, N., Patwardhan, M.S. (2018). Mental health analysis using deep learning for feature extraction. In Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, Goa, India, pp. 356-359. https://doi.org/10.1145/3152494.3167990

[14] Calvaresi, D., Calbimonte, J.P., Siboni, E., Eggenschwiler, S., Manzo, G., Hilfiker, R., Schumacher, M. (2021). EREBOTS: Privacy-compliant agent-based platform for multi-scenario personalized health-assistant chatbots. Electronics, 10(6): 666. https://doi.org/10.3390/electronics10060666

[15] Low, Y.S., Jackson, M.L., Hyde, R.J., et al. (2024). Answering real-world clinical questions using large language model based systems. arXiv preprint arXiv:2407.00541. https://doi.org/10.48550/arXiv.2407.00541

[16] Bhat, V., Cheerla, S.D., Mathew, J.R., Pathak, N., Liu, G., Gao, J. (2024). Retrieval augmented generation (rag) based restaurant chatbot with AI testability. In 2024 IEEE 10th International Conference on Big Data Computing Service and Machine Learning Applications (BigDataService), Shanghai, China, pp. 1-10. https://doi.org/10.1109/BigDataService62917.2024.00008

[17] Duan, Z., Wang, J. (2024). Exploration of LLM multi-agent application implementation based on langgraph+ crewai. arXiv preprint arXiv:2411.18241. https://doi.org/10.48550/arXiv.2411.18241

[18] Acharya, D.B., Kuppan, K., Divya, B. (2025). Agentic AI: Autonomous intelligence for complex goals—A comprehensive survey. IEEE Access, 13: 18912-18936. https://doi.org/10.1109/ACCESS.2025.3532853

[19] Hosseini, S., Seilani, H. (2025). The role of agentic AI in shaping a smart future: A systematic review. Array, 26: 100399. https://doi.org/10.1016/j.array.2025.100399

[20] Singh, A., Ehtesham, A., Kumar, S., Khoei, T.T. (2025). Agentic retrieval-augmented generation: A survey on agentic rag. arXiv preprint arXiv:2501.09136. https://doi.org/10.48550/arXiv.2501.09136

[21] Xu, C., Jiang, X. (2022). iMedBot: A web-based intelligent agent for healthcare related prediction and deep learning. arXiv preprint arXiv:2210.05671. https://doi.org/10.48550/arXiv.2210.05671

[22] Schmidgall, S., Ziaei, R., Harris, C., Reis, E., Jopling, J., Moor, M. (2024). Agentclinic: A multimodal agent benchmark to evaluate ai in simulated clinical environments. arXiv preprint arXiv:2405.07960. https://doi.org/10.48550/arXiv.2405.07960